CE246/IT257: Database Management System

Dec – May 2020-21

Database Security

Devang Patel Institute of Advance Technology and Research

What is View

- Views are virtual only and run the query definition each time they are accessed.
- It is a logical view of your tables, with no data stored anywhere else.

- Views are essentially logical table-like structures populated on the fly by a given query. The results of a view query are not stored anywhere on disk and the view is recreated every time the query is executed

- Create view Syntax:

  CREATE VIEW view_name AS SELECT columns FROM table WHERE conditions;

  Example: create view a_view as select * from a_student;

View

What is a View?

EMPLOYEES Table:

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALA
100	Steven	King	SKING	515.123.4567	17-JUN-87	AD_PRES	240
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-89	AD_VP	170
102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-93	AD_VP	170
103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-90	IT_PROG	90
104	Bruce	Ernst	BERNST	590.423.4568	21-MAY-91	IT_PROG	60
107	Diana	Lorentz	DLORENTZ	590.423.5567	07-FEB-99	IT_PROG	42
124	Kevin	Mourgos	KMOURGOS	650.123.5234	16-NOV-99	ST_MAN	58
141	Trenna	Rajs	TRAJS	650.121.8009	17-OCT-95	ST_CLERK	35
142	Curtis	Davies	CDAVIES	650.121.2994	29-JAN-97	ST_CLERK	31
143	Randall	Matos	RMATOS	650.121.2874	15-MAR-98	ST_CLERK	26
					JUL-98	ST_CLERK	25
					JAN-00	SA_MAN	105
					MAY-96	SA_REP	110
					MAR-98	SA_REP	86
178	Kimberely	Grant	KGRANT	011.44.1644.429263	24-MAY-99	SA_REP	70
200	Jennifer	Whalen	JWHALEN	515.123.4444	17-SEP-87	AD_ASST	44
201	Michael	Hartstein	MHARTSTE	515.123.5555	17-FEB-96	MK_MAN	130
202	Pat	Fay	PFAY	603.123.6666	17-AUG-97	MK_REP	60
205	Shelley	Higgins	SHIGGINS	515.123.8080	07-JUN-94	AC_MGR	120
206	William	Gietz	WGIETZ	515.123.8181	07-JUN-94	AC_ACCOUNT	83

20 rows selected.

EMPLOYEE_ID	LAST_NAME	SALARY
149	Zlotkey	10500
174	Abel	11000
176	Taylor	8600

View Retrieval/ Update

- This Oracle CREATE VIEW example would create a virtual table based on the result set of the SELECT statement. You can now query the Oracle VIEW as follows:

  select * from a_view;

- You can modify the definition of an Oracle VIEW without dropping it by using the Oracle CREATE OR REPLACE VIEW Statement

  CREATE OR REPLACE VIEW view_name AS SELECT columns FROM table WHERE conditions;

- create or replace will update view if view exists, if doesn't then it creates a view

  Example: create or replace view a_view1 as select sid from a_student where sid>=102;

Drop View

- To drop a created view
- drop view view_name;

  Example: drop view a_view1;

Question: Can you update the data in an Oracle VIEW?

Answer: A VIEW in Oracle is created by joining one or more tables. When you update record(s) in a VIEW, it updates the records in the underlying tables that make up the view. So, yes, you can update the data in an Oracle VIEW providing you have the proper privileges to the underlying Oracle tables.

Question: Does the Oracle View exist if the table is dropped from the database?

Answer: Yes, in Oracle, the VIEW continues to exist even after one of the tables (that the Oracle VIEW is based on) is dropped from the database. However, if you try to query the Oracle VIEW after the table has been dropped, you will receive a message indicating that the Oracle VIEW has errors.

If you recreate the table (the table that you had dropped), the Oracle VIEW will again be fine.

Materialized View

- Materialized view (MV) is indirect access to table data by storing the result of a query in a separate schema object.
- It can be stored in the same db or in different db.
- MV stored in same db as their master tables can improve performance through query rewrite. For queries that involve aggregate data or join the optimizer can rewrite the query to access the precomputed results stored in MV.
- Query rewrite is particularly useful in data warehouse.
- Another name of mv is snapshot.
- MV are schema object that is used to summarize, precompute, replicate, distribute data.
- Suitable in data warehouse, decision support, distributed & mobile computing.
- They consume storage
- Must be refreshed when data in master table changes.
- Their existent is transparent to sql application & user.
- Having limitation: support for incremental refresh.

Differentiate view and materialized view

view	Materialized view
are virtual only and run the query definition each time they are accessed	are disk based and update periodically base upon the query definition.
Views evaluate the data in the tables underlying the view definition at the time the view is queried	
It is a logical view of your tables, with no data stored anywhere else.	Materialized views are similar to regular views
The upside of a view is that it will always return the latest data to you.	The upside of this is this is that when you query a materialized view, you are querying a table, which may also be indexed
	with query rewrite enabled, Oracle can optimize a query that selects from the source of your materialized view in such a way that it instead reads from your materialized view

The downside of a view is that its performance depends on how good a select statement the view is based on. If the select statement used by the view joins many tables, or uses joins based on non-indexed columns, the view could perform poorly.	**The downside though is that the data you get back from the materialized view is only as up to date as the last time the materialized view has been refreshed.**
	Materialized views can be set to refresh manually, on a set schedule, or based on the database detecting a change in data from one of the underlying tables.
	Materialized views are most often used in data warehousing / business intelligence applications where querying large fact tables with thousands of millions of rows would result in query response times that resulted in an unusable application.
Views are essentially logical table-like structures populated on the fly by a given query. The results of a view query are not stored anywhere on disk and the view is recreated every time the query is executed	Materialized views are actual structures stored within the database and written to disk. They are updated based on the parameters defined when they are created.

Limitation & Syntax of Materialized View

- Support for incremental refresh is limited.
- **Syntax**

```
CREATE MATERIALIZED VIEW view-name
    BUILD [IMMEDIATE | DEFERRED]
    REFRESH [FAST | COMPLETE | FORCE ]
    ON [COMMIT | DEMAND ]
    [[ENABLE | DISABLE] QUERY REWRITE]
    [ON PREBUILT TABLE]
        AS
        SELECT ...;
```

- The BUILD clause options are shown below.

  IMMEDIATE : The materialized view is populated immediately.

  DEFERRED : The materialized view is populated on the first requested refresh.

- The following refresh types are available.

  FAST : A fast refresh is attempted. If materialized view logs are not present against the source tables in advance, the creation fails.

  COMPLETE : The table segment supporting the materialized view is truncated and repopulated completely using the associated query.

  FORCE : A fast refresh is attempted. If one is not possible a complete refresh is performed.

- A refresh can be triggered in one of two ways.

  ON COMMIT : The refresh is triggered by a committed data change in one of the dependent tables.

  ON DEMAND : The refresh is initiated by a manual request or a scheduled task.

- The **QUERY REWRITE** clause tells the optimizer if the materialized view should be consider for query rewrite operations.
- The **ON PREBUILT TABLE** clause tells the database to use an existing table segment, which must have the same name as the materialized view and support the same column structure as the query

Create materialized view

- Example: create materialized view a_mv1 build immediate refresh fast on commit as select sid from a_student;

Error: ORA-01031: insufficient privileges, on a_student

It's easier to GRANT or REVOKE privileges to the users through a role rather than assigning a privilege directly to every user. If a role is identified by a password, then, when you GRANT or REVOKE privileges to the role, you definitely have to identify it with the password.

Authentication vs Authorization

Authentication	Authorization
Authentication verifies who you are	Authorization verifies what you are authorized to do.
For example, you can login	For example, you are allowed to login into your Unix server via ssh client, but you are not authorized to browser /data2 or any other file system.
	Authorization occurs after successful authentication
	Authorization can be controlled at file system level or using various application level configuration
Authentication: I am an employee of the company. Here is my ID badge.	Authorization: As an employee of the company, I am allowed entrance into the building.

Authorization:

- Privileges
  - **System privileges**

    Each system privilege allows a user to perform a particular database operation or class of database operations; for example, the privilege to create tablespaces is a system privilege.
  - **Object privileges**

    Each object privilege allows a user to perform a particular action on a specific object, such as a table, view, sequence, procedure, function, or package

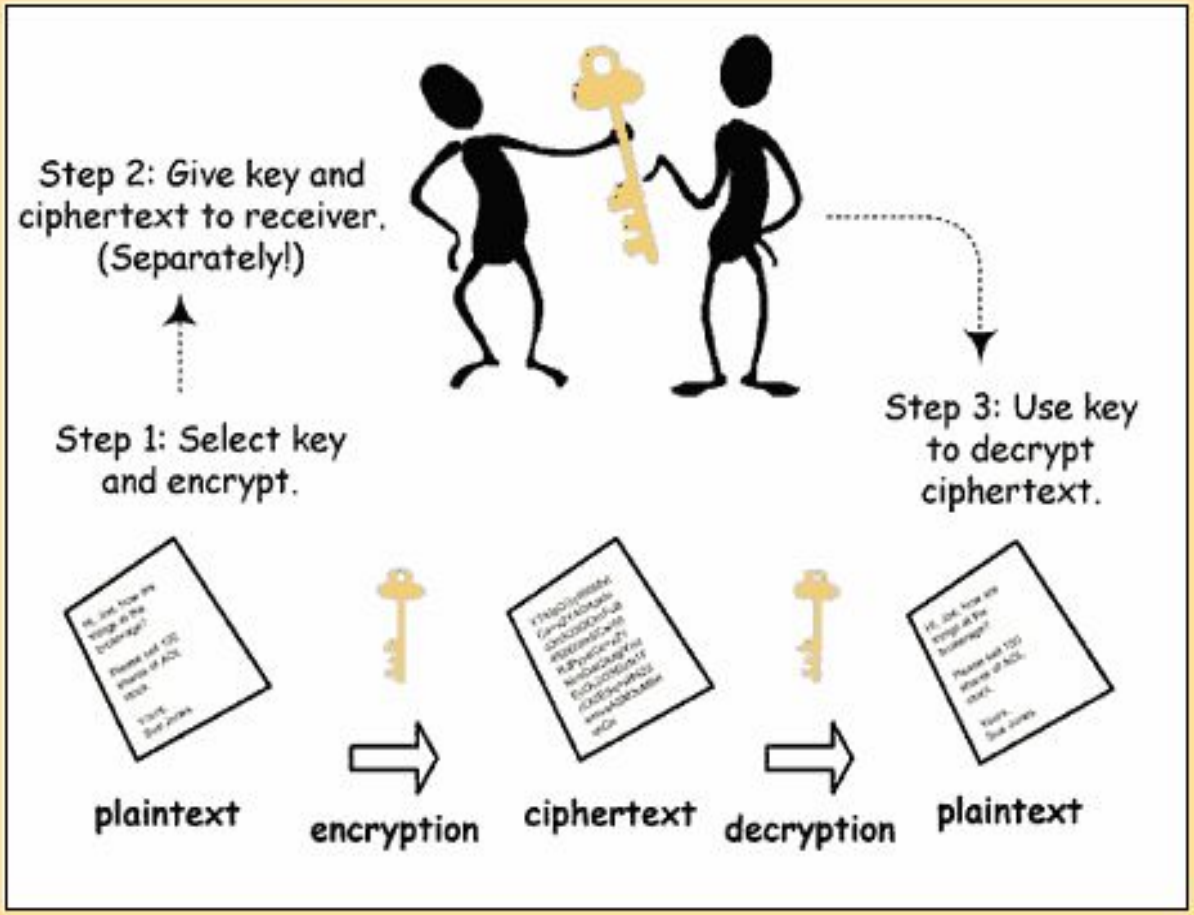
- Role

Data Encryption

- Data encryption is the act of changing electronic information into an unreadable state by using algorithms or ciphers.
- Over time as the public has begun to enter and transmit personal, sensitive information over the internet, data encryption has become more widespread

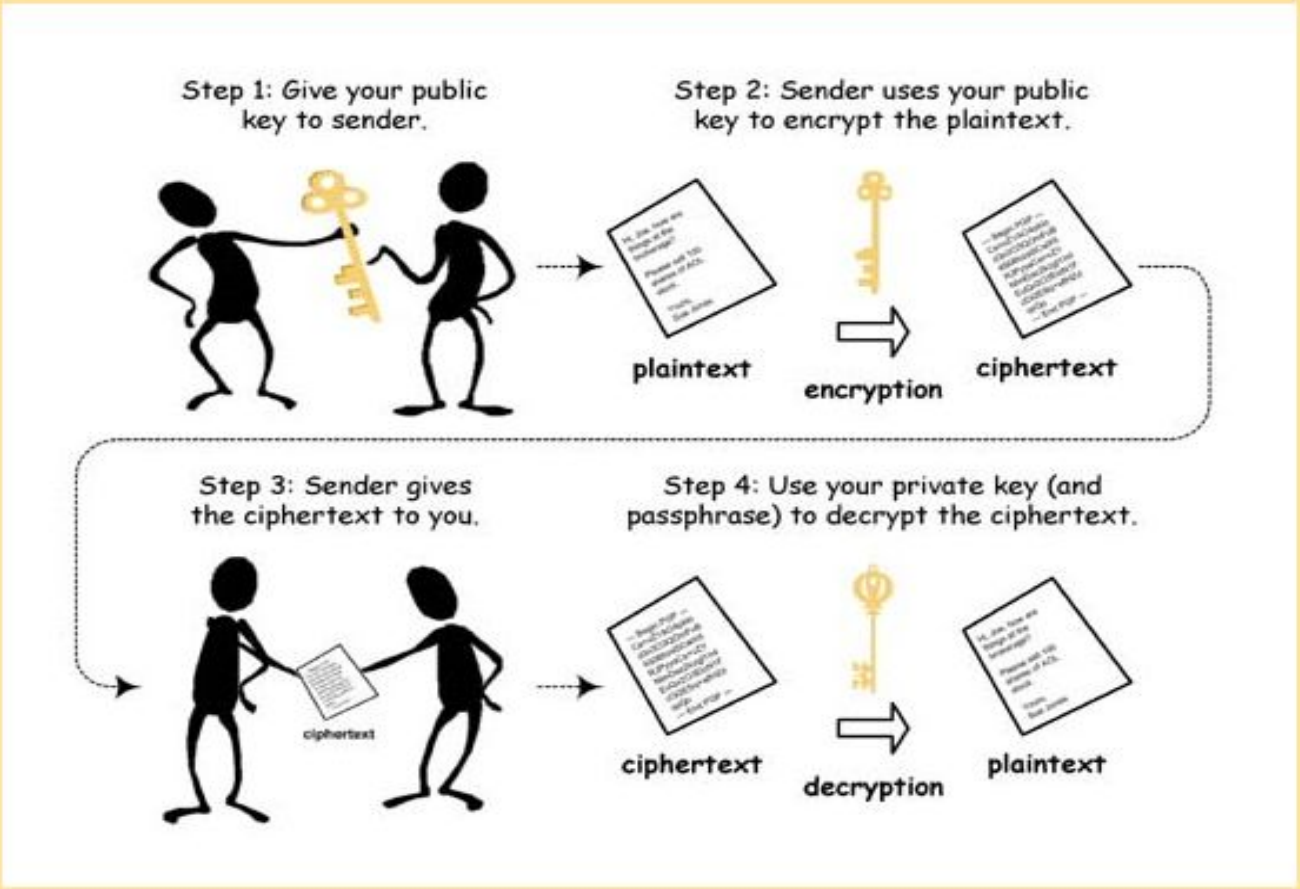
Symmetric-key Cryptography

Symmetric-key cryptography refers to encryption methods in which both the sender and receiver share the same key.



Asymmetric-key Cryptography

Public-key is a type of asymmetric key cryptography in which two different keys are used. There is a public key and a private key. A public key system is sort of like a mail slot - any one can **encrypt** (put mail in your mail slot), but only the holder of the private key can **decrypt** (the owner of the mail box who has its key).



RSA

- The RSA algorithm, introduced in 1977 by Rivest, Shamir, and Adlemen, is an algorithm for public-key cryptography.
- RSA was the first and is still the most widely-used algorithm for public key cryptography and it is used for thousands of applications from e-mail encryption to secure online purchasing.
- It was the first cryptosystem to enable senders to “sign” each message they send so that the recipient has proof of who sent the message.

RSA (Choosing Public / Private Key)

- Pick two large (100 digit) primes ***p*** and ***q***.
- Let $n = p*q$
- Select a relatively small integer ***e*** that is prime to Phi $\phi(n) = (p-1)(q-1)$,
  $1 < e < \phi(n)$
- Find *d*, the multiplicative inverse of e that is relatively prime to the Phi
  That is inverse of public key
  $[e* d \bmod [\phi(n) = 1]$

RSA …..

- (e,n) is the public key. To encrypt M, compute
  - En(M) Cipher = $M^e \pmod n$
- (d,n) is the private key. To decrypt C, compute
  - De(C) Pt = $C^d \pmod n$

- Let p = 11, q = 5
- n = pq = 55
- phi(n)= (p-1)(q-1) = 40 == { 3, 7 , 9, 11…}
- Possible e:, … (let’s use 7)
- Find d: 7*d (mod(40)) = 1 .(apply Euclidean algo.)
- Public key: (7, 55)
- Private key: (23, 55)
- Message=3
- En(42) = 3^7 (mod 55) = 42
- De(81) = 42^23(mod 55) = 3

- Let p = 11, q = 13
- n = pq = 143
- phi(n)= (p-1)(q-1) = 120 = 3 x 23 x 5
- Possible e: 7, 11, 13, 17, … (let’s use 7)
- Find d: 7*d = 1(mod 120) = 103 e*d=1 mod phi(n)
- Public key: (7, 143)
- Private key: (103, 143)
- Message=h=7
- En(42) = 7^7 (mod 143) = 6
- De(81) = 6^103(mod 143) = 7

Strengths of RSA

- No prior communication needed
- Highly secure (for large enough keys)
- Well-understood
- Allows both encryption and signing

- The security of the RSA algorithm and messages encrypted using the algorithm relies on the difficulty of factoring the value of n. If n could be easily factored into the corresponding values of p and q, then one could easily find the value of d.
- The RSA Assumption is that the RSA Problem is hard to solve when n is sufficiently large and randomly generated.

Missing Information

ID	Name	Marital Status	Salary
1	A	Married	
2	B		15000
3		Unmarried	23000

Sometimes we don’t know what value an entry in a relation should have

- We know that there is a value, but don’t know what it is
- There is no value at all that makes any sense

Two main methods have been proposed to deal with this

- NULLs can be used as markers to show that information is missing
- A default value can be used to represent the missing value

NULL's

- NULL is a placeholder for missing or unknown value of an attribute. It is not itself a value.
- Codd proposed to distinguish two kinds of NULLs:
  - A-marks: data Applicable but not known (for example, someone's age)
  - I-marks: data is Inapplicable (telephone number for someone who does not have a telephone, or spouse's name for someone who is not married)

Problems with NULLs

- Additional problems for SQL: do we treat NULLs as duplicates? Do we include them in count, sum, average and if yes, how? How do arithmetic operations behave when an argument is NULL?

Theoretical solutions

- Use three-valued logic instead of classical two-valued logic to evaluate conditions.
- When there are no NULLs around, conditions evaluate to true or false, but if a null is involved, a condition will evaluate to the third value (‘undefined’, or ‘unknown’ or ‘unk’).

- Default values are an alternative to the use of NULLs
  - If a value is not known a particular placeholder value - the default - is used
  - These are actual values, so don’t need 3VL etc.

Boolean Operators

and	t	u	f
T	t	u	F
U	t	u	F
f	f	f	f

or	t	u	f
T	t	t	t
U	t	u	u
f	t	u	f

not	
T	f
U	u
f	t

3 Value Logic-3VL

X	Y	X AND Y	X OR Y	NOT X
T	**T**	T	T	F
T	**U**	U	T	F
T	**F**	F	T	F
U	**T**	U	T	U
U	**U**	U	U	U
U	**F**	F	U	U
F	**T**	F	T	T
F	**U**	F	U	T
F	**F**	F	F	T

For example A=3, B=4 and C=unk

Check

1. a>b and b>c= 3>4 and 4>unk=f and unk=f
2. a>b or b>c=3>4 or 4 > unk=f or unk=unk
3. A<b or b<c=3<4 or 4<unk=t or unk=t
4. not(A=c)=not(3=unk)=not(unk)=unk

Maybe Boolean Operator

Maybe	
T	f
U	t
f	f

Exists expression

- If r is a relation with tuples t(1),t(2),…,t(m)
- V is a range variable that range over r
- P(v) is a Boolean expression in which v occurs as a free variable
- Then the expression:::

Exists V(p(v)) is defined to be equivalent to False or (p(t1) or p(t2) or… or p(tm))

ID	Name	Marital Status	Salary
1	A	Married	
2	B		15000
3		Unmarried	23000

Exists (Marital_status=Married)
=f or (t or unk or f)
=f or t
=t

Exists (Maybe((Marital_status=Married)))
=f or (maybe(t) or maybe(unk) or maybe(f))
=f or (f or t or f)
=f or t
=true

ForAll expression

- If r is a relation with tuples t(1),t(2),…,t(m)
- V is a range variable that range over r
- P(v) is a Boolean expression in which v occurs as a free variable
- Then the expression:::

ForAll V(p(v)) is defined to be equivalent to true and (p(t1) and p(t2) and… and p(tm))

ID	Name	Marital Status	Salary
1	A	Married	
2	B		15000
3		Unmarried	23000

Forall (Marital_status=Married)
=t and (t and unk and f)
=t and f
=f

IS_UNK operator

It takes a single scaler operand and returns true if that operand evaluates to unk otherwise false

ID	Name	Marital Status	Salary
1	A	Married	
2	B		15000
3		Unmarried	23000

Exists (IS_UNK(name)
=f or (f or f or t)
=t

Forall(IS_UNK(name)
=t and (f and f and t)
=f

IF_UNK operator

IF_UNK operator takes 2 scaler operands and return the value of 1st operand unless that operand evaluates to unk

IF_UNK(exp1,exp2)= IF_UNK(exp1) then return exp2 else exp1;

Note: exp1 & exp2 must be of same type

Example:::: IF_UNK k(name,'no name')

Whenever name is unknown 'no name' will be the default value

UNK== unk ?

UNK= the value unknown

unk=the unknown truth value

X is a boolean variable can have values like true, false or unk.

“x is unk”= value of x is known to be unk

“x is UNK”= value of x is not known

Example1

STU_ID	STU_NAME	SCHOLARSHIP_AMOUNT	ELECTIVE_SUBJECT
1	Samantha	2000	WT
2	Smith		.NET
3	David	2000	.NET
4			WT
5	Jennifer		
6		2000	.NET

1. EXISTS V(V. SCHOLARSHIP_AMOUNT=unk)
2. FORALL V((V.STU_ID!=3) && (V.ELECTIVE_SUBJECT !=.NET))
3. EXISTS V (IS_UNK(V.ELECTIVE_SUBJECT))
4. FORALL V(V.STU_ID <10 OR V.STU_ID>=0)
5. EXISTS V(MAYBE (IS_UNK(V.STU_NAME)))

Solution1

1. EXISTS V(V. SCHOLARSHIP_AMOUNT=unk)= F 0R (F OR T OR F OR T OR T OR F)=T
2. FORALL V((V.STU_ID!=3) && (V.ELECTIVE_SUBJECT !=.NET))= T AND ((T AND T) AND (T AND F) AND (F AND F) AND (T AND T) AND (T AND UNK) AND (T AND F))= T AND T AND F AND F AND T AND UNK AND F =F
3. EXISTS V (IS_UNK(V.ELECTIVE_SUBJECT))=F OR F OR F OR F OR F OR T OR F =T
4. FORALL V(V.STU_ID <10 OR V.STU_ID>=0)= T AND ((T OR T)AND (T OR T) AND (T OR T) AND (T OR T) AND (T OR T) AND (T OR T))=T AND T AND T AND T AND T AND T AND T =T
5. EXISTS V(MAYBE (IS_UNK(V.STU_NAME)))=F OR (MB(F) OR MB(F) OR MB(F) OR MB(T) OR MB(F) OR MB(T))=F OR F OR F OR F OR F OR F OR F =F

Example2

Branch_Id	Branch_Name	City	Emp_Head
B001	KALUPUR	AHMEDABAD	Anil
B002	KAROLI BAUG	VADODRA	Dhani
B003	MALAD	Unk	Unk
B005	Unk	Unk	Mukesh

a) EXISTS(maybe(V.Branch_id=B003))
b) FORALL V (maybe(city=unk) or is_unk(Emp_head))
c) EXISTS V(is_unk(Branch_Name))

Example: inner join

Student

ID	Name
123	John
124	Mary
125	Mark
126	Jane

Enrolment

ID	Code	Mark
123	DBS	60
124	PRG	70
125	DBS	50
128	DBS	80

← dangles

Student inner join Enrolment

ID	Name	ID	Code	Mark
123	John	123	DBS	60
124	Mary	124	PRG	70
125	Mark	125	DBS	50

Outer Join

- When we take the join of two relations we match up tuples which share values
  - Some tuples have no match, and are ‘lost’ these are called ‘dangles’
- Outer joins include dangles in the result and use NULLs to fill in the blanks
  - Left outer join
  - Right outer join
  - Full outer join

Example: left outer join

Student

ID	Name
123	John
124	Mary
125	Mark
126	Jane

Enrolment

ID	Code	Mark
123	DBS	60
124	PRG	70
125	DBS	50
128	DBS	80

← dangles

Student left outer join Enrolment

ID	Name	ID	Code	Mark
123	John	123	DBS	60
124	Mary	124	PRG	70
125	Mark	125	DBS	50
126	Jane	null	null	null

Example: right outer join

Student

ID	Name
123	John
124	Mary
125	Mark
126	Jane

Enrolment

ID	Code	Mark
123	DBS	60
124	PRG	70
125	DBS	50
128	DBS	80

← dangles

Student right outer join Enrolment

ID	Name	ID	Code	Mark
123	John	123	DBS	60
124	Mary	124	PRG	70
125	Mark	125	DBS	50
null	null	128	DBS	80

Example: full outer join

Student

ID	Name
123	John
124	Mary
125	Mark
126	Jane

Enrolment

ID	Code	Mark
123	DBS	60
124	PRG	70
125	DBS	50
128	DBS	80

← dangles

Student full outer join Enrolment

ID	Name	ID	Code	Mark
123	John	123	DBS	60
124	Mary	124	PRG	70
125	Mark	125	DBS	50
126	Jane	null	null	null
null	null	128	DBS	80

Missing Information

Outer Join Syntax

SELECT <cols> FROM <table1> <type> OUTER JOIN <table2> ON <condition>

Example:

SELECT * FROM Student FULL OUTER JOIN Enrolment ON

Student.ID = Enrolment.ID